

CSE 390a Lecture 3

bash shell continued:
processes; multi-user systems; remote login; editors

slides created by Marty Stepp, modified by Jessica Miller and Ruth Anderson
<http://www.cs.washington.edu/390a/>

Lecture summary

- A bit more on combining commands
- Processes and basic process management
- Connecting to remote servers (attu)
 - multi-user environments
- Text editors

Review: Redirection and Pipes

- **`command > filename`**
 - Write the output of **`command`** to **`filename`** (> to append instead)
- **`command < filename`**
 - Use **`filename`** as the input stream to **`command`**
- **`command1 | command2`**
 - Use the console output of **`command1`** as the input to **`command2`**
- **`command1 ; command2`**
 - Run **`command1`** and then run **`command2`**
- **`command1 && command2`**
 - Run **`command1`**, if completed without errors then run **`command2`**

Tricky Examples

- The `wc` command can take multiple files: `wc names.txt student.txt`
 - Can we use the following to `wc` on every `txt` file in the directory?
 - `ls *.txt | wc`
- Amongst the top 250 movies in `movies.txt`, display the third to last movie that contains "The" in the title when movies titles are sorted.
- Find the disk space usage of the `man` program
 - Hints: use `which` and `du`...
 - Does `which man | du` work?

The back-tick

`command1 `command2``

- run **`command2`** and pass its console output to **`command1`** as a parameter; ``` is a back-tick, on the `~` key; not an apostrophe
- best used when **`command2`**'s output is short (one line)
- Finish the example!
 - `du `which man``

xargs

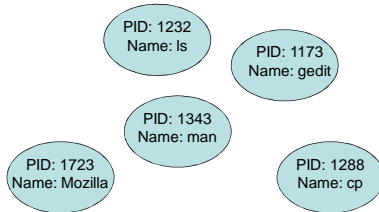
command	description
<code>xargs</code>	run each line of input as an argument to a specified command

- `xargs` allows you to repeatedly run a command over a set of lines
 - often used in conjunction with `find` to process each of a set of files
- Example: Remove all my `.class` files.


```
find ~ -name "*.class" | xargs rm
```
- Find the disk usage of `man` using `xargs`
 - `which man | xargs du`

Processes

- **process**: a program that is running (essentially)
 - when you run commands in a shell, it launches a process for each command
 - Process management is one of the major purposes of an OS



Process commands

command	description
ps or jobs	list processes being run by a user; each process has a unique integer id (PID)
top	show which processes are using CPU/memory; also shows stats about the computer
kill	terminate a process by PID
killall	terminate several processes by name

- use **kill** or **killall** to stop a runaway process (infinite loop)
 - similar to ^C hotkey, but doesn't require keyboard intervention

Background processes

command	description
&	(special character) when placed at the end of a command, runs that command in the background
^Z	(hotkey) suspends the currently running process
fg, bg	resumes the currently suspended process in either the foreground or background

- If you run a graphical program like **gedit** from the shell, the shell will lock up waiting for the graphical program to finish
 - instead, run the program in the background, so the shell won't wait: `$ gedit resume.txt &`
 - if you forget to use **&**, suspend **gedit** with **^Z**, then run **bg**
 - lets play around with an infinite process...

Connecting with ssh

command	description
ssh	open a shell on a remote server

- Linux/Unix are built to be used in multi-user environments where several users are logged in to the same machine at the same time
 - users can be logged in either locally or via the network
- You can connect to other Linux/Unix servers with **ssh**
 - once connected, you can run commands on the remote server
 - other users might also be connected; you can interact with them
 - can connect even from other operating systems

The attu server

- **attu** : The UW CSE department's shared Linux server
- connect to **attu** by typing:


```
ssh attu.cs.washington.edu
```

(or `ssh username@attu.cs.washington.edu` if your Linux system's user name is different than your CSE user name)
- Note: There are several computers that respond as **attu** (to spread load), so if you want to be on the same machine as your friend, you may need to connect to **attu2**, **attu3**, etc.

Multi-user environments

command	description
whoami	outputs your username
passwd	changes your password
hostname	outputs this computer's name/address
w or finger	see info about people logged in to this server
write	send a message to another logged in user

- *Exercise* : Connect to **attu**, and send somebody else a message.

Network commands

command	description
links or lynx	text-only web browsers (really!)
ssh	connect to a remote server
sftp or scp	transfer files to/from a remote server (after starting sftp, use get and put commands)
wget	download from a URL to a file
curl	download from a URL and output to console
alpine, mail	text-only email programs

13

Text editors

command	description
pico or nano	simple but crappy text editors (recommended)
emacs	complicated text editor
vi or vim	complicated text editor

- you cannot run graphical programs when connected to `attu` (yet)
 - so if you want to edit documents, you need to use a text-only editor
- most advanced Unix/Linux users learn `emacs` or `vi`
 - these editors are powerful but complicated and hard to learn
 - we recommend the simpler `nano` (hotkeys are shown on screen)

14

Mounting remote files

command	description
sshfs	mount and interact with remote directories and files

- An alternate usage model to remotely connecting to servers is mounting remote directories and files and work on them locally
 - once mounted, use remote directories and files as if they were local
- To mount a remote directory
 - create a local directory to mount to
`mkdir csehomedir`
 - mount your remote files on your local system
`sshfs username@attu.cs.washington.edu:/homes/iws/username csehomedir/`

15

Aliases

command	description
alias	assigns a pseudonym to a command

`alias name=command`

- must wrap the command in quotes if it contains spaces
- Example: When I type `q`, I want it to log me out of my shell.
- Example: When I type `ll`, I want it to list all files in long format.


```
alias q=exit
alias ll="ls -la"
```
- *Exercise*: Make it so that typing `q` quits out of a shell.
- *Exercise*: Make it so that typing `woman` runs `man`.
- *Exercise*: Make it so that typing `attu` connects me to `attu`.

16